Building end-user applications on embedded devices with WPE

Mario Sánchez Prada

mario@igalia.com

igalia

About me

- **CS Engineer**, partner of **Igalia**
- **Involvement** in some **Open Source communities**
  - e.g. Chromium, WebKit, GNOME
- Other **work** done in the past:
  - Linux-based OS's (i.e. Endless OS, Litl OS)
  - Maemo (Hildon Application Manager)
  - Samsung SmartTV platform

Currently **coordinating** Igalia's **WebKit team**

About Igalia

- Specialized **Open Source consultancy**, founded in 2001
- **Fully remote**, HQ in **A Coruña** (Spain). **Flat structure**.
- **Top contributors** to all the main **Web Rendering Engines**
  - WebKit, Chromium, Gecko and Servo
- **Active contributor to other areas and OSS projects**
  - V8, SpiderMonkey, JSC, LLVM, Node.js, GStreamer, Mesa, Linux Kernel...
- **Members of several working groups**:
  - W3C, WHATWG, WPT, TC39, OpenJS, Test262, Khronos...

igalia

Open Source Consultancy

Outline

- Web rendering engines
- What is WebKit?
- What is WPE
- Integrating WPE in your product
- Conclusion

Web rendering engines

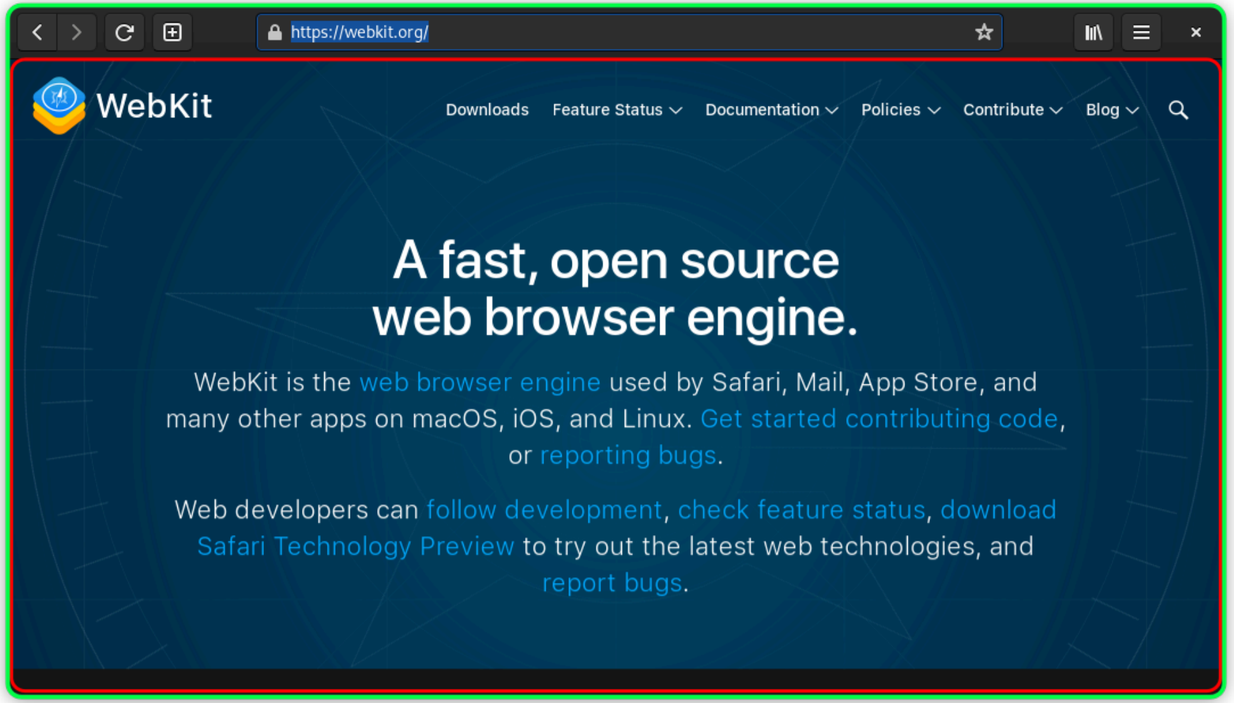
(aka *Web engines*)

What is a Web rendering engine?

A Web rendering engine is the **software component** that enables you to leverage the power of the **Web Platform in your product**:

- **Fetchs** HTML/CSS/JavaScript content from multiple sources.
- **Interprets** the Web content to produce an internal representation.
- **Produces a result** that users can **interact with**.
- It's a **extremely flexible platform**. Some examples:
  - Textual and non-textual content
  - Multimedia playback
  - Fully-fledge applications

What is a Web rendering engine?



Web rendering **engine** ≠ Web **browser**

Most popular Web engines

servo

What is WebKit?

What is WebKit?

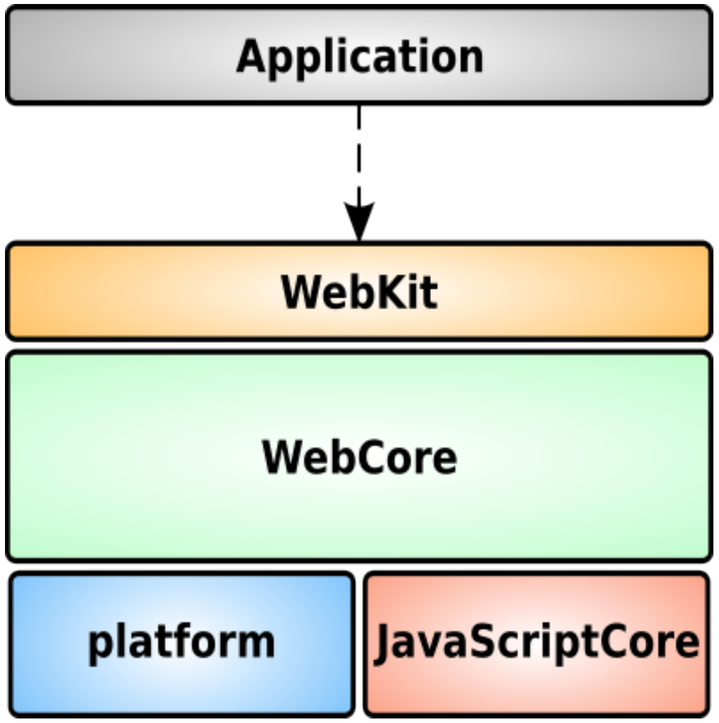
- Open Source **Web rendering engine**
  - Started as a fork of KHTML and KJS in 2001.
  - Forked again become *Blink* in 2013.
- **Main goals**:
  - Performance, portability, stability, compatibility, standards compliance, security, *hackability* and ***embeddability***.
- Support for **different platforms**:
  - **Desktop & Mobile**: Mac, iOS and Linux
  - **Embedded**: set-top-boxes, video game consoles, smart home appliances, in-vehicle/inflight entertainment, GPS devices, digital signage...

Advantages of WebKit

- 🕸️ **Complete implementation** of the Web Platform
- 🔌 **Embeddable** as top priority
- 🧩 **Flexible and modular** architecture
- 🔒 **Privacy** and **security** by design
- 🚀 **Performance** and **stability**
- 🐧 **Independent** Linux-based flavours
  - 👌 Not controlled by any big corporation

WebKit Architecture

- **Application**:
  - What end-users interact with.
- **WebKit**:
  - Exposes an API to applications and implements the split-process model.
- **WebCore**:
  - Layout, rendering, network, multimedia, accessibility...
- **JavaScriptCore**:
  - The JavaScript engine.
- **Platform**:
  - Platform-specific hooks.



WebKit Ports

- **WebKit port**: adaptation of WebKit to a specific platform.

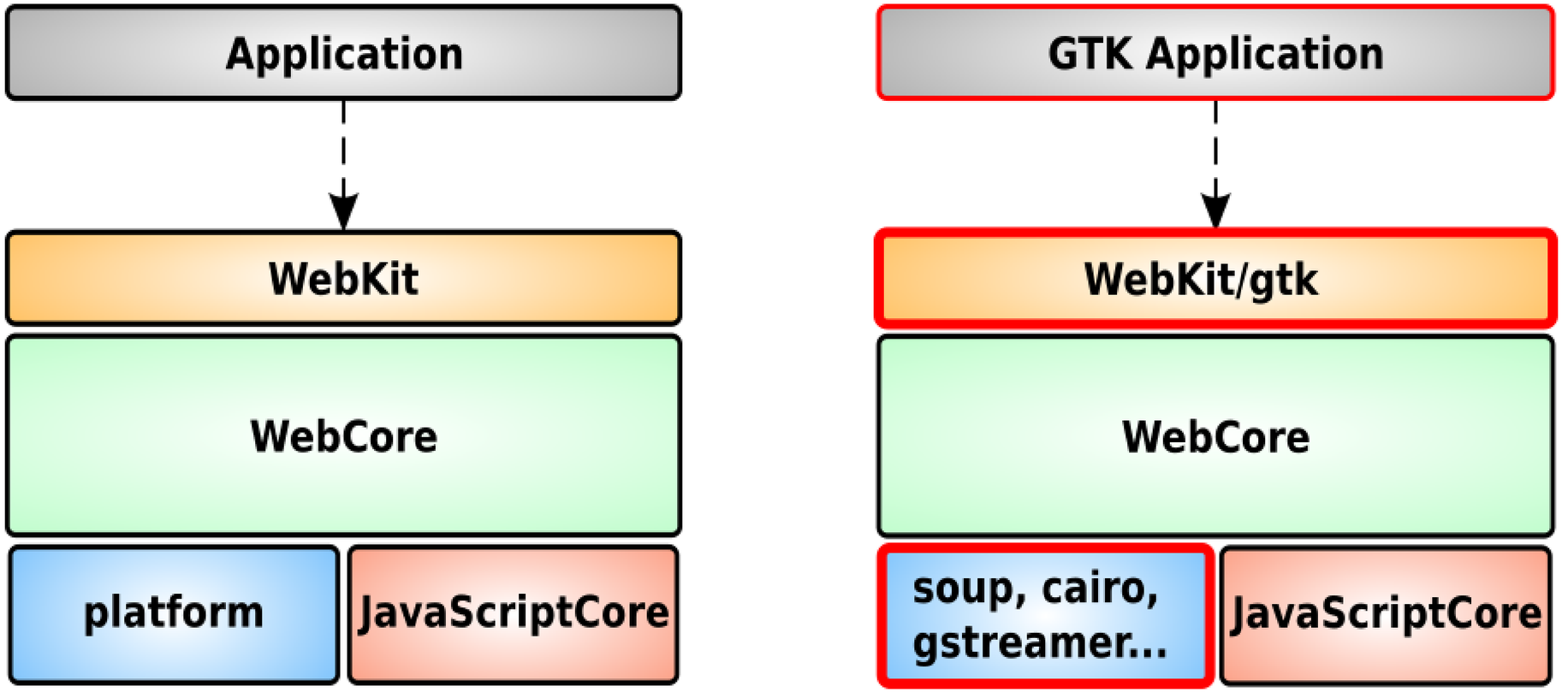
- **Official WebKit Ports** (*upstream* ports)
  - **Mac**: Safari, Apple Mail, iTunes, App Store...
  - **iOS**: every browser on iOS devices (including Chrome).
  - **WinCairo**: Microsoft Playwright, Playstation SDK
  - **Playstation**: Playstation s4 & Playstation 5
  - **WebKitGTK**: GNOME Web, Evolution, Shotwell...
  - **WPE**: Cog and other custom-made "browsers" for embedded devices.

https://docs.webkit.org/Ports/Introduction.html

WebKitGTK and WPE

- **WebKit ports** targeting **Linux-based** systems
  - **Common parts**: GLib, libsoup (networking), GStreamer (multimedia)...
  - **Key differences**: graphics stack, input handling. Different use cases.
- **WebKitGTK**:
  - Go-to solution to embed Web content in GTK applications.
  - Integration with GNOME components. Supports GTK3 and GTK4.
- **WPE**:
  - Lower level, aimed at embedded devices.
  - Requires graphics and input backend to work.

WebKit Ports: WebKitGTK



What is WPE?

What is WPE?

WebKit **port optimized** for Linux **embedded devices**

- Fully operational **JavaScript engine** (64-bit, also 32bit).
- Focus on **flexibility**, **security** and **performance**.
- **Minimal** set of **dependencies**
- **Backends-based** architecture for input and output.
- **Low memory** and **storage footprint**.
- Support for **HW-accelerated graphics** and **multimedia**.

ℹ️ **https://wpewebkit.org/**

What is not WPE?

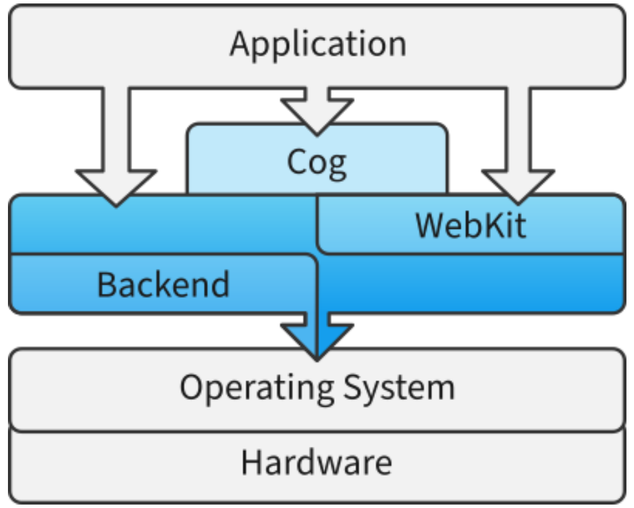
***NOT** a general purpose Web Browser*

- Provides just the **building block**s for Web-based applications.
- Doesn't implement all the **APIs** found on other WebKit ports.
- Does not rely on any particular **UI Toolkit** (i.e. backends).
- Can also be useful for **less conventional use cases**

    e.g. server-side rendering, headless mode...

WPE Architecture

- **Application**:
  - The end application, which can use WPE directly or via the provided Cog launcher.
- **WebKit**:
  - The actual WebKit port, including the API layer to link against from applications.
- **Backend**:
  - Platform-specific implementation of the graphics and/or the input layers.



WPE components

- **WPEWebKit**:
  - The actual WebKit port.
  - Relies on the backends for page display and input.
- **libwpe**:
  - Provides rendering-related callbacks implemented by the graphical backend.
  - Allows the input backend to rely events from the application to WebKit.
- **WPEBackend-FDO**:
  - The reference FreekDesktop.Org-based backend (i.e. Wayland).
  - Supports several architectures plus regular PC architectures.
  - Can be replaced by a device-specific backend
- **Cog**:
  - Small single “window” launcher for WPE, with no user interface.

WPE-based products

- Some **examples of use cases** we are aware about:
  - Set-Top-Boxes (both *RDK* and non *RDK* based)
  - Smart Home Appliances
  - HiFi audio/sound systems & music streaming
  - Digital Signage
  - GPS navigation devices
  - Video/Audio conference
  - Headless server-side rendering
  - QA and testing
  - ...

Demos

tdl.js - spacerocks

fps: 33
Meteors
1
a few
many
lots

100
80
60
40
20
0
cpu: 17.45
fps: 34.34

igalia

Integrating WPE in your project

1. Carefully choose your hardware

- Determine the **specifics of the HW** you'll rely on.
  - It might impose specific constraints (e.g. linear-only direct scanout)
- Determine the **input devices** needed:
  - e.g. Physical keyboards, touch-based devices
- Determine the **output devices** needed:
  - e.g. Standard screens, custom displays, no displays at all
- Figure out **whether custom backends** are needed
  - Otherwise WPE provides Wayland and DRM already

2. Assemble the required components

- **WPE libraries**:
  - **Current** API: WPEWebKit, libwpe, backends
  - **New** API: WPEWebKit, WPEPlatform (includes built-in platforms)
- **WPE dependencies**:
  - 2D/3D rendering: cairo, OpenGL, EGL, GLES...
  - Multimedia pipeline: GStreamer
  - Networking: libsoup
  - Other: glib, atk, wayland...
- **Other libraries**:
  - Anything else your application needs
- **Launcher application**:
  - Cog (uses the old API)
  - Your own application

Example application (new API)

```c
#include <wpe/webkit.h>

int main(int argc, const char *argv[]) {
 g_autoptr(GMainLoop) loop = g_main_loop_new(NULL, FALSE);
 g_autoptr(WebKitWebView) view = webkit_web_view_new(NULL);

 webkit_web_view_load_uri(view,
  (argc > 1) ? argv[1] : "https://wpewebkit.org");

 g_main_loop_run(loop);

 return EXIT_SUCCESS;
}
```


3. Develop your product

- **Prototype/Write your Web application**:
  - Plain HTML/CSS/JS and/or common frameworks (e.g. React, Angular, Vue...)
  - Prototype your application (e.g. on desktop device)
- **Avoid poorly performing features on your platform**:
  - Certain constrained devices might impose specific limitations
  - Ideally avoid particularly expensive operations (e.g. blur)
- **Integrate with the rest of your platform**:
  - Leverage the Web Platform APIs to implement most features
  - Possible to extend functionality through custom adaptation
- **Testing & QA**:
  - Run tests on target HW (e.g. i.MX6, i.MX8, Broadcom...)
  - Testing automation frameworks: WebDriver, PlayWright

4. Bundle up and ship

- **Distribution packages available**:
  - **Debian**: **https://packages.debian.org/source/stable/wpewebkit**
  - **Raspbian**: **https://archive.raspbian.org/raspbian/pool/main/w/wpewebkit**
  - **Arch Linux**: **https://archlinux.org/packages/extra/x86_64/wpewebkit**
- **Source code**:
  - **Tarballs** available at **https://wpewebkit.org/release**
  - **Git repositories**:
    - WebKit: **https://github.com/webKit/WebKit/**
    - LibWPE: **https://github.com/WebPlatformForEmbedded/libwpe**
    - FDO Backend: **https://github.com/Igalia/WPEBackend-fdo**
    - Cog: **https://github.com/Igalia/cog**

4. Bundle up and ship

- **Yocto-based images**:
  - **Layer** available at **https://github.com/Igalia/meta-webkit**
  - Building an image with WPE: **https://github.com/Igalia/meta-webkit/wiki/WPE**
  - **Recommended** way of building WPE-based images
- **Buildroot-based images**:
  - Support available at:
    - **https://github.com/buildroot/buildroot/blob/master/package/wpewebkit/wpewebkit.mk**
    - **https://github.com/buildroot/buildroot/blob/master/package/libwpe/libwpe.mk**
    - **https://github.com/buildroot/buildroot/blob/master/package/wpebackend-fdo/wpebackend-fdo.mk**
    - **https://github.com/buildroot/buildroot/blob/master/package/cog/cog.mk**
  - Updated on a **best-effort** basis

Last but not least: collaborate upstream!

- **WPE WebKit website**: **https://wpewebkit.org**
- **API documentation**: **https://wpewebkit.org/reference/stable**
- **Security Advisories**: **https://wpewebkit.org/security**
- **Mailing list**: **https://lists.webkit.org/mailman/listinfo/webkit-wpe**
- **Matrix**: **https://matrix.to/#/#wpe:matrix.org**
- **Bug tracker**: **https://bugs.webkit.org**
  - See also **https://webkit.org/reporting-bugs**

Conclusion

Conclusion

- **Web Rendering Engines** useful for more than Web Browsers.
- **WPE** leverages WebKit's strengths and makes it ideal for embedded devices, providing the **extra flexibility** needed.
- **Integrating WPE** in your project requires carefully **choosing your Hardware** and assembling the **required components**.
- Products can be developed as **regular Web applications**, and **adaptations are still possible** thanks to WPE's modular design.
- WPE and WebKit **fully Open Source**, contributions are welcome!

Questions?

Thanks!


igalia